KPMG Virtual Experience

1. Data_Quality_Assessment

Transaction

In [1]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
```

In [2]:
```
import scipy.stats as stats
```

In [3]:
```
# read in the excel dataset and put into a pandas datafra
data=pd.ExcelFile("KPMG_VI_New_raw_data_update_final.xlsx")
pd1=pd.read_excel(data,sheet_name=1,header=1)
```

In [4]:
```
df1=pd.DataFrame(pd1)
```

In [5]:
```
df1
```

Out[5]:

	transaction_id	product_id	customer_id	transaction_date	online_order	order_status	brand	product_line	product_class	product_size	list_price	standard_cost	product_first_sold_date
0	1	2	2950	2017-02-25	0.0	Approved	Solex	Standard	medium	medium	71.49	53.62	41245.0
1	2	3	3120	2017-05-21	1.0	Approved	Trek Bicycles	Standard	medium	large	2091.47	388.92	41701.0
2	3	37	402	2017-10-16	0.0	Approved	OHM Cycles	Standard	low	medium	1793.43	248.82	36361.0
3	4	88	3135	2017-08-31	0.0	Approved	Norco Bicycles	Standard	medium	medium	1198.46	381.10	36145.0
4	5	78	787	2017-10-01	1.0	Approved	Giant Bicycles	Standard	medium	large	1765.30	709.48	42226.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
19995	19996	51	1018	2017-06-24	1.0	Approved	OHM Cycles	Standard	high	medium	2005.66	1203.40	37823.0
19996	19997	41	127	2017-11-09	1.0	Approved	Solex	Road	medium	medium	416.98	312.74	35560.0
19997	19998	87	2284	2017-04-14	1.0	Approved	OHM Cycles	Standard	medium	medium	1636.90	44.71	40410.0
19998	19999	6	2764	2017-07-03	0.0	Approved	OHM Cycles	Standard	high	medium	227.88	136.73	38216.0
19999	20000	11	1144	2017-09-22	1.0	Approved	Trek Bicycles	Standard	medium	small	1775.81	1580.47	36334.0

20000 rows × 13 columns

In [6]:
```
df1.shape
```

Out[6]:
```
(20000, 13)
```

In [7]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype         
---  ------                   --------------  -----         
 0   transaction_id           20000 non-null  int64         
 1   product_id               20000 non-null  int64         
 2   customer_id              20000 non-null  int64         
 3   transaction_date         20000 non-null  datetime64[ns]
 4   online_order             19640 non-null  float64       
 5   order_status             20000 non-null  object        
 6   brand                    19803 non-null  object        
 7   product_line             19803 non-null  object        
 8   product_class            19803 non-null  object        
 9   product_size             19803 non-null  object        
 10  list_price               20000 non-null  float64       
 11  standard_cost            19803 non-null  float64       
 12  product_first_sold_date  19803 non-null  float64       
dtypes: datetime64[ns](1), float64(4), int64(3), object(5)
memory usage: 2.0+ MB
```

In [8]:
```
df1.describe()
```

Out[8]:

	transaction_id	product_id	customer_id	online_order	list_price	standard_cost	product_first_sold_date
count	20000.000000	20000.00000	20000.000000	19640.000000	20000.000000	19803.000000	19803.000000
mean	10000.500000	45.36465	1738.246050	0.500458	1107.829449	556.046951	38199.776549
std	5773.647028	30.75359	1011.951046	0.500013	582.825242	405.955660	2875.201110
min	1.000000	0.00000	1.000000	0.000000	12.010000	7.210000	33259.000000
25%	5000.750000	18.00000	857.750000	0.000000	575.270000	215.140000	35667.000000
50%	10000.500000	44.00000	1736.000000	1.000000	1163.890000	507.580000	38216.000000
75%	15000.250000	72.00000	2613.000000	1.000000	1635.300000	795.100000	40672.000000
max	20000.000000	100.00000	5034.000000	1.000000	2091.470000	1759.850000	42710.000000

In [9]:
```
df1.dtypes
```

Out[9]:
```
transaction_id                    int64
product_id                        int64
customer_id                       int64
transaction_date         datetime64[ns]
online_order                    float64
order_status                     object
brand                            object
product_line                     object
product_class                    object
product_size                     object
list_price                      float64
standard_cost                   float64
product_first_sold_date         float64
dtype: object
```

In [10]:
```
df1.isnull().sum()
```

Out[10]:
```
transaction_id             0
product_id                 0
customer_id                0
transaction_date           0
online_order             360
order_status               0
brand                    197
product_line             197
product_class            197
product_size             197
list_price                 0
standard_cost            197
product_first_sold_date  197
dtype: int64
```

In [11]:
```
for col in df1.columns:
    print(f"Features Name : {col}\n {df1[col].unique()}\n\n")
    print(f"Features Name : {col}\n  {df1[col].value_counts()}\n\n")
```

```
Features Name : transaction_id
 [    1     2     3 ... 19998 19999 20000]


Features Name : transaction_id
  1       1
13331    1
13338    1
13337    1
13336    1
        ..
6667     1
6666     1
6665     1
6664     1
20000    1
Name: transaction_id, Length: 20000, dtype: int64


Features Name : product_id
 [  2   3  37  88  78  25  22  15  67  12   5  61  35  16  79  33  54  27
  82  89  64  19  72  91   1  99   0  92  14  44  76  46  55  66  81  86
  32  77  96   6  47  94  93  60  28   4  38  56  58  50  80  87  84  21
  31  62  17  73  45  49  95  18  70  26  39  36  98  75  42  20  24  53
  65  29  11  10   7  41   9  69  90  97 100  74  71  34  57  23  51  59
  63  40   8  13  30  48  68  83  43  52  85]


Features Name : product_id
  0      1378
3       354
1       311
35      268
38      267
       ...
71      137
8       136
16      136
100     130
47      121
Name: product_id, Length: 101, dtype: int64


Features Name : customer_id
 [2950 3120  402 ...  130 2789 3446]


Features Name : customer_id
  2183    14
2476    14
1068    14
1672    13
2912    13
        ..
898      1
2352     1
1846     1
3279     1
1757     1
Name: customer_id, Length: 3494, dtype: int64


Features Name : transaction_date
 ['2017-02-25T00:00:00.000000000' '2017-05-21T00:00:00.000000000'
 '2017-10-16T00:00:00.000000000' '2017-08-31T00:00:00.000000000'
 '2017-10-01T00:00:00.000000000' '2017-03-08T00:00:00.000000000'
 '2017-04-21T00:00:00.000000000' '2017-07-15T00:00:00.000000000'
 '2017-08-10T00:00:00.000000000' '2017-08-30T00:00:00.000000000'
 '2017-01-17T00:00:00.000000000' '2017-01-05T00:00:00.000000000'
 '2017-02-26T00:00:00.000000000' '2017-09-10T00:00:00.000000000'
 '2017-06-11T00:00:00.000000000' '2017-10-10T00:00:00.000000000'
 '2017-04-03T00:00:00.000000000' '2017-06-02T00:00:00.000000000'
 '2017-04-06T00:00:00.000000000' '2017-01-28T00:00:00.000000000'
 '2017-10-09T00:00:00.000000000' '2017-06-29T00:00:00.000000000'
 '2017-04-08T00:00:00.000000000' '2017-10-18T00:00:00.000000000'
 '2017-01-10T00:00:00.000000000' '2017-04-11T00:00:00.000000000'
 '2017-12-23T00:00:00.000000000' '2017-10-13T00:00:00.000000000'
 '2017-03-15T00:00:00.000000000' '2017-09-05T00:00:00.000000000'
 '2017-02-18T00:00:00.000000000' '2017-03-20T00:00:00.000000000'
 '2017-02-28T00:00:00.000000000' '2017-08-20T00:00:00.000000000'
 '2017-07-07T00:00:00.000000000' '2017-01-09T00:00:00.000000000'
 '2017-12-06T00:00:00.000000000' '2017-09-12T00:00:00.000000000'
 '2017-11-28T00:00:00.000000000' '2017-05-08T00:00:00.000000000'
 '2017-05-14T00:00:00.000000000' '2017-03-17T00:00:00.000000000'
 '2017-12-22T00:00:00.000000000' '2017-06-07T00:00:00.000000000'
 '2017-06-20T00:00:00.000000000' '2017-05-07T00:00:00.000000000'
 '2017-06-10T00:00:00.000000000' '2017-06-17T00:00:00.000000000'
 '2017-05-10T00:00:00.000000000' '2017-03-24T00:00:00.000000000'
 '2017-11-27T00:00:00.000000000' '2017-12-21T00:00:00.000000000'
 '2017-02-09T00:00:00.000000000' '2017-09-18T00:00:00.000000000'
 '2017-11-14T00:00:00.000000000' '2017-02-15T00:00:00.000000000'
 '2017-12-18T00:00:00.000000000' '2017-02-24T00:00:00.000000000'
 '2017-12-17T00:00:00.000000000' '2017-07-08T00:00:00.000000000'
 '2017-05-16T00:00:00.000000000' '2017-02-23T00:00:00.000000000'
 '2017-07-03T00:00:00.000000000' '2017-09-30T00:00:00.000000000'
 '2017-10-21T00:00:00.000000000' '2017-05-06T00:00:00.000000000'
 '2017-01-03T00:00:00.000000000' '2017-05-09T00:00:00.000000000'
 '2017-07-23T00:00:00.000000000' '2017-01-21T00:00:00.000000000'
 '2017-03-10T00:00:00.000000000' '2017-11-07T00:00:00.000000000'
 '2017-12-16T00:00:00.000000000' '2017-10-07T00:00:00.000000000'
 '2017-04-27T00:00:00.000000000' '2017-04-01T00:00:00.000000000'
 '2017-11-02T00:00:00.000000000' '2017-02-13T00:00:00.000000000'
 '2017-07-19T00:00:00.000000000' '2017-05-23T00:00:00.000000000'
 '2017-08-06T00:00:00.000000000' '2017-04-04T00:00:00.000000000'
 '2017-08-15T00:00:00.000000000' '2017-04-15T00:00:00.000000000'
 '2017-07-25T00:00:00.000000000' '2017-02-21T00:00:00.000000000'
 '2017-05-31T00:00:00.000000000' '2017-06-23T00:00:00.000000000'
 '2017-03-26T00:00:00.000000000' '2017-07-31T00:00:00.000000000'
 '2017-07-27T00:00:00.000000000' '2017-04-14T00:00:00.000000000'
 '2017-11-12T00:00:00.000000000' '2017-10-26T00:00:00.000000000'
 '2017-10-11T00:00:00.000000000' '2017-03-27T00:00:00.000000000'
 '2017-11-04T00:00:00.000000000' '2017-10-24T00:00:00.000000000'
 '2017-05-01T00:00:00.000000000' '2017-11-21T00:00:00.000000000'
 '2017-03-21T00:00:00.000000000' '2017-02-20T00:00:00.000000000'
 '2017-06-16T00:00:00.000000000' '2017-12-04T00:00:00.000000000'
 '2017-12-01T00:00:00.000000000' '2017-03-05T00:00:00.000000000'
 '2017-03-16T00:00:00.000000000' '2017-06-05T00:00:00.000000000'
 '2017-09-04T00:00:00.000000000' '2017-01-18T00:00:00.000000000'
 '2017-09-28T00:00:00.000000000' '2017-09-02T00:00:00.000000000'
 '2017-10-17T00:00:00.000000000' '2017-04-25T00:00:00.000000000'
 '2017-09-23T00:00:00.000000000' '2017-08-02T00:00:00.000000000'
 '2017-03-25T00:00:00.000000000' '2017-01-01T00:00:00.000000000'
 '2017-12-25T00:00:00.000000000' '2017-01-26T00:00:00.000000000'
 '2017-12-20T00:00:00.000000000' '2017-01-13T00:00:00.000000000'
 '2017-11-11T00:00:00.000000000' '2017-08-19T00:00:00.000000000'
 '2017-08-22T00:00:00.000000000' '2017-09-15T00:00:00.000000000'
 '2017-03-01T00:00:00.000000000' '2017-08-16T00:00:00.000000000'
 '2017-08-27T00:00:00.000000000' '2017-07-10T00:00:00.000000000'
 '2017-12-11T00:00:00.000000000' '2017-05-20T00:00:00.000000000'
 '2017-07-30T00:00:00.000000000' '2017-10-25T00:00:00.000000000'
 '2017-04-26T00:00:00.000000000' '2017-02-02T00:00:00.000000000'
 '2017-05-02T00:00:00.000000000' '2017-10-22T00:00:00.000000000'
 '2017-11-16T00:00:00.000000000' '2017-03-30T00:00:00.000000000'
 '2017-04-13T00:00:00.000000000' '2017-01-31T00:00:00.000000000'
 '2017-03-18T00:00:00.000000000' '2017-01-07T00:00:00.000000000'
 '2017-07-05T00:00:00.000000000' '2017-11-19T00:00:00.000000000'
 '2017-11-08T00:00:00.000000000' '2017-07-24T00:00:00.000000000'
 '2017-01-15T00:00:00.000000000' '2017-01-12T00:00:00.000000000'
 '2017-06-30T00:00:00.000000000' '2017-06-22T00:00:00.000000000'
 '2017-12-27T00:00:00.000000000' '2017-06-08T00:00:00.000000000'
 '2017-07-29T00:00:00.000000000' '2017-05-27T00:00:00.000000000'
 '2017-05-05T00:00:00.000000000' '2017-09-09T00:00:00.000000000'
 '2017-09-22T00:00:00.000000000' '2017-04-12T00:00:00.000000000'
 '2017-12-14T00:00:00.000000000' '2017-03-22T00:00:00.000000000'
 '2017-11-29T00:00:00.000000000' '2017-06-09T00:00:00.000000000'
 '2017-11-13T00:00:00.000000000' '2017-04-09T00:00:00.000000000'
 '2017-11-30T00:00:00.000000000' '2017-03-13T00:00:00.000000000'
 '2017-08-12T00:00:00.000000000' '2017-08-04T00:00:00.000000000'
 '2017-09-01T00:00:00.000000000' '2017-07-02T00:00:00.000000000'
 '2017-03-11T00:00:00.000000000' '2017-10-08T00:00:00.000000000'
 '2017-11-23T00:00:00.000000000' '2017-01-22T00:00:00.000000000'
 '2017-01-29T00:00:00.000000000' '2017-07-21T00:00:00.000000000'
 '2017-02-10T00:00:00.000000000' '2017-11-25T00:00:00.000000000'
 '2017-12-03T00:00:00.000000000' '2017-12-30T00:00:00.000000000'
 '2017-07-26T00:00:00.000000000' '2017-06-15T00:00:00.000000000'
 '2017-07-06T00:00:00.000000000' '2017-12-28T00:00:00.000000000'
 '2017-01-30T00:00:00.000000000' '2017-06-26T00:00:00.000000000'
 '2017-08-13T00:00:00.000000000' '2017-01-25T00:00:00.000000000'
 '2017-07-11T00:00:00.000000000' '2017-04-30T00:00:00.000000000'
 '2017-05-19T00:00:00.000000000' '2017-05-26T00:00:00.000000000'
```

```
 '2017-02-03T00:00:00.000000000' '2017-04-28T00:00:00.000000000'
 '2017-03-23T00:00:00.000000000' '2017-06-03T00:00:00.000000000'
 '2017-05-04T00:00:00.000000000' '2017-03-29T00:00:00.000000000'
 '2017-10-03T00:00:00.000000000' '2017-08-07T00:00:00.000000000'
 '2017-02-07T00:00:00.000000000' '2017-11-20T00:00:00.000000000'
 '2017-04-05T00:00:00.000000000' '2017-10-28T00:00:00.000000000'
 '2017-08-25T00:00:00.000000000' '2017-01-20T00:00:00.000000000'
 '2017-03-14T00:00:00.000000000' '2017-05-24T00:00:00.000000000'
 '2017-11-18T00:00:00.000000000' '2017-04-07T00:00:00.000000000'
 '2017-01-04T00:00:00.000000000' '2017-12-13T00:00:00.000000000'
 '2017-04-20T00:00:00.000000000' '2017-03-07T00:00:00.000000000'
 '2017-08-01T00:00:00.000000000' '2017-06-19T00:00:00.000000000'
 '2017-01-23T00:00:00.000000000' '2017-04-19T00:00:00.000000000'
 '2017-08-11T00:00:00.000000000' '2017-11-15T00:00:00.000000000'
 '2017-11-01T00:00:00.000000000' '2017-10-05T00:00:00.000000000'
 '2017-10-30T00:00:00.000000000' '2017-01-02T00:00:00.000000000'
 '2017-07-09T00:00:00.000000000' '2017-10-20T00:00:00.000000000'
 '2017-11-10T00:00:00.000000000' '2017-08-14T00:00:00.000000000'
 '2017-09-13T00:00:00.000000000' '2017-05-03T00:00:00.000000000'
 '2017-01-24T00:00:00.000000000' '2017-08-03T00:00:00.000000000'
 '2017-08-28T00:00:00.000000000' '2017-06-24T00:00:00.000000000'
 '2017-10-27T00:00:00.000000000' '2017-03-03T00:00:00.000000000'
 '2017-12-02T00:00:00.000000000' '2017-10-15T00:00:00.000000000'
 '2017-10-12T00:00:00.000000000' '2017-03-19T00:00:00.000000000'
 '2017-06-27T00:00:00.000000000' '2017-02-27T00:00:00.000000000'
 '2017-03-12T00:00:00.000000000' '2017-06-14T00:00:00.000000000'
 '2017-02-17T00:00:00.000000000' '2017-09-14T00:00:00.000000000'
 '2017-04-17T00:00:00.000000000' '2017-05-11T00:00:00.000000000'
 '2017-11-03T00:00:00.000000000' '2017-10-02T00:00:00.000000000'
 '2017-12-10T00:00:00.000000000' '2017-05-25T00:00:00.000000000'
 '2017-08-26T00:00:00.000000000' '2017-09-06T00:00:00.000000000'
 '2017-04-24T00:00:00.000000000' '2017-12-15T00:00:00.000000000'
 '2017-02-12T00:00:00.000000000' '2017-07-22T00:00:00.000000000'
 '2017-09-26T00:00:00.000000000' '2017-12-05T00:00:00.000000000'
 '2017-03-31T00:00:00.000000000' '2017-11-22T00:00:00.000000000'
 '2017-04-29T00:00:00.000000000' '2017-07-14T00:00:00.000000000'
 '2017-09-08T00:00:00.000000000' '2017-04-22T00:00:00.000000000'
 '2017-01-11T00:00:00.000000000' '2017-08-09T00:00:00.000000000'
 '2017-05-29T00:00:00.000000000' '2017-06-13T00:00:00.000000000'
 '2017-07-28T00:00:00.000000000' '2017-04-23T00:00:00.000000000'
 '2017-07-04T00:00:00.000000000' '2017-10-19T00:00:00.000000000'
 '2017-09-21T00:00:00.000000000' '2017-04-02T00:00:00.000000000'
 '2017-08-18T00:00:00.000000000' '2017-08-21T00:00:00.000000000'
 '2017-03-06T00:00:00.000000000' '2017-01-16T00:00:00.000000000'
 '2017-02-22T00:00:00.000000000' '2017-03-09T00:00:00.000000000'
 '2017-01-27T00:00:00.000000000' '2017-12-19T00:00:00.000000000'
 '2017-04-16T00:00:00.000000000' '2017-10-04T00:00:00.000000000'
 '2017-02-16T00:00:00.000000000' '2017-08-17T00:00:00.000000000'
 '2017-02-01T00:00:00.000000000' '2017-12-08T00:00:00.000000000'
 '2017-08-23T00:00:00.000000000' '2017-05-12T00:00:00.000000000'
 '2017-03-04T00:00:00.000000000' '2017-05-28T00:00:00.000000000'
 '2017-07-20T00:00:00.000000000' '2017-07-13T00:00:00.000000000'
 '2017-10-14T00:00:00.000000000' '2017-06-12T00:00:00.000000000'
 '2017-02-14T00:00:00.000000000' '2017-02-11T00:00:00.000000000'
 '2017-10-23T00:00:00.000000000' '2017-06-28T00:00:00.000000000'
 '2017-06-04T00:00:00.000000000' '2017-04-18T00:00:00.000000000'
 '2017-09-16T00:00:00.000000000' '2017-05-15T00:00:00.000000000'
 '2017-05-22T00:00:00.000000000' '2017-06-06T00:00:00.000000000'
 '2017-12-09T00:00:00.000000000' '2017-07-18T00:00:00.000000000'
 '2017-02-06T00:00:00.000000000' '2017-11-09T00:00:00.000000000'
 '2017-06-21T00:00:00.000000000' '2017-07-17T00:00:00.000000000'
 '2017-10-29T00:00:00.000000000' '2017-05-13T00:00:00.000000000'
 '2017-10-06T00:00:00.000000000' '2017-02-04T00:00:00.000000000'
 '2017-11-05T00:00:00.000000000' '2017-02-08T00:00:00.000000000'
 '2017-11-06T00:00:00.000000000' '2017-09-20T00:00:00.000000000'
 '2017-04-10T00:00:00.000000000' '2017-11-17T00:00:00.000000000'
 '2017-12-07T00:00:00.000000000' '2017-12-12T00:00:00.000000000'
 '2017-01-19T00:00:00.000000000' '2017-05-17T00:00:00.000000000'
 '2017-02-19T00:00:00.000000000' '2017-09-19T00:00:00.000000000'
 '2017-09-29T00:00:00.000000000' '2017-05-18T00:00:00.000000000'
 '2017-07-01T00:00:00.000000000' '2017-12-29T00:00:00.000000000'
 '2017-11-24T00:00:00.000000000' '2017-01-08T00:00:00.000000000'
 '2017-08-24T00:00:00.000000000' '2017-07-12T00:00:00.000000000'
 '2017-08-08T00:00:00.000000000' '2017-02-05T00:00:00.000000000'
 '2017-07-16T00:00:00.000000000' '2017-09-27T00:00:00.000000000'
 '2017-01-14T00:00:00.000000000' '2017-03-28T00:00:00.000000000'
 '2017-08-29T00:00:00.000000000' '2017-09-24T00:00:00.000000000'
 '2017-10-31T00:00:00.000000000' '2017-12-24T00:00:00.000000000'
 '2017-09-03T00:00:00.000000000' '2017-09-17T00:00:00.000000000'
 '2017-08-05T00:00:00.000000000' '2017-06-18T00:00:00.000000000'
 '2017-09-07T00:00:00.000000000' '2017-09-25T00:00:00.000000000'
 '2017-06-01T00:00:00.000000000' '2017-11-26T00:00:00.000000000'
 '2017-09-11T00:00:00.000000000' '2017-01-06T00:00:00.000000000'
 '2017-12-26T00:00:00.000000000' '2017-03-02T00:00:00.000000000'
 '2017-05-30T00:00:00.000000000' '2017-06-25T00:00:00.000000000']


Features Name : transaction_date
  2017-02-14    82
2017-08-18    82
2017-10-15    76
2017-01-31    73
2017-12-19    71
              ..
2017-01-12    38
2017-12-07    37
2017-03-29    36
2017-09-25    35
2017-10-19    32
Name: transaction_date, Length: 364, dtype: int64


Features Name : online_order
 [ 0.  1. nan]


Features Name : online_order
  1.0    9829
0.0    9811
Name: online_order, dtype: int64


Features Name : order_status
 ['Approved' 'Cancelled']


Features Name : order_status
  Approved     19821
Cancelled      179
Name: order_status, dtype: int64


Features Name : brand
 ['Solex' 'Trek Bicycles' 'OHM Cycles' 'Norco Bicycles' 'Giant Bicycles'
 'WeareA2B' nan]


Features Name : brand
  Solex            4253
Giant Bicycles    3312
WeareA2B          3295
OHM Cycles        3043
Trek Bicycles     2990
Norco Bicycles    2910
Name: brand, dtype: int64


Features Name : product_line
 ['Standard' 'Road' 'Mountain' 'Touring' nan]


Features Name : product_line
  Standard    14176
Road         3970
Touring      1234
Mountain      423
Name: product_line, dtype: int64


Features Name : product_class
 ['medium' 'low' 'high' nan]


Features Name : product_class
  medium    13826
high       3013
low        2964
Name: product_class, dtype: int64
```

```
Features Name : product_size
 ['medium' 'large' 'small' nan]


Features Name : product_size
  medium    12990
large      3976
small      2837
Name: product_size, dtype: int64


Features Name : list_price
 [  71.49 2091.47 1793.43 1198.46 1765.3  1538.99   60.34 1292.84 1071.23
 1231.15  574.64   71.16 1057.51 1661.92 1555.58 1311.44  499.53 1362.99
 1469.44  360.4   642.31 1403.5  1720.7   544.05 1415.01 1842.92 1769.64
 2083.94 1289.85 1894.19 1163.89 1151.96  235.63  642.7  1240.31 1635.3
  227.88  363.01  100.35 1458.17 1977.36   12.01 1216.14 1129.13  183.86
  912.52  175.89 1073.07 1179.    958.74  792.9   290.62  752.64  478.16
 1024.66 1945.43  441.49  533.51  569.56 1148.64  495.72 1992.93 1812.75
  945.04  358.39 1873.97 1810.   1775.81 1777.8   795.34  575.27 1172.78
 1065.03 1807.45 1942.61 1274.93 1890.39  980.37  416.98 1386.84  742.54
  230.91  688.63  748.17 1466.68 1656.86  202.62 1036.59 1228.07  774.53
  586.45 1762.96 2005.66 1483.2   590.26 1703.52 1577.53 1636.9  1280.28
 1061.56  850.89  710.59  543.39 1227.34 1972.01  311.54  205.84  605.54
  756.31  850.    571.27  294.35  149.3   447.25 1348.41 1021.43 1163.77
 1294.66  638.29 1202.34  326.86  814.86 1102.68 1880.4  1541.62  639.38
  402.14  710.55 1142.89  813.6  1833.17 1285.47 1678.71  437.51  366.22
 1199.26 1888.45  237.44  317.3  1527.25 1563.94 1146.42 1192.33 1672.07
 1099.68 1281.6  1220.66 1061.92 1502.93  398.08  101.55 1957.87 1095.66
 1776.08  752.36  172.09 1740.16 1734.3   240.72 1958.34  330.36 1709.26
   60.72 1193.44 2037.77 1349.47   56.21  126.36 1300.96 1697.27   26.15
  875.99 1862.82 2028.26  678.25  890.28 1084.18 1107.08 1790.31  724.37
 1095.65 1223.24  541.44  470.07 2052.92 1817.13 1473.09 1480.76 2012.84
 1711.35  753.76  840.02 1922.94 1759.3   330.91 1029.36 1487.8  1624.69
 1719.95  899.52  554.38  960.51 1978.64   36.78   74.89 1091.51  695.23
 1148.41  750.25 1997.68 1392.76 1294.37  311.57  899.92  162.87 1678.51
 1703.18  675.28  487.8  1292.13 1578.52  922.15  122.74  435.66 1409.68
 1315.16 1233.12  758.73  405.94 1026.7   687.9   494.96 1181.89  465.71
  880.3  1989.3  1413.98  483.12  753.01  883.91 1216.4    32.44   16.08
 1184.    587.02 1919.23 1660.68 1224.41 1535.84  150.51 2062.95 1689.63
 1934.41  889.05 1617.32 1788.93 2064.08 2020.14  489.58 2076.81 2061.38
  877.44 1374.2   731.41 1034.17  156.56  643.95  517.53  226.94 1196.65
  634.57 1571.83 1999.54 1562.88 1150.59  356.05 1780.22 1438.9   966.91
  918.43 1541.1  1476.88 1023.39  310.57  356.39 1695.42  877.6  1401.63
 2086.07 1761.75 1615.62  744.54 1098.18  868.56 1497.43  867.92]


Features Name : list_price
  2091.47    465
1403.50      396
71.49        274
1231.15      235
1890.39      233
            ...
56.21          1
126.36         1
1300.96        1
1697.27        1
867.92         1
Name: list_price, Length: 296, dtype: int64


Features Name : standard_cost
 [  53.62       388.92       248.82       381.1        709.48
  829.65        45.26        13.44       380.74       161.6
  459.71        56.93       154.4       1479.11       818.01
 1167.18       388.72        57.74       596.55       270.3
  513.85       954.82      1531.42       376.84      1259.36
 1105.75       108.76       675.03        74.51       598.76
  589.27       649.49       125.07       211.37       795.1
  993.66       136.73       290.41        75.26       874.9
 1759.85         7.21      1082.36       677.48       137.9
  141.4        131.92       933.84       707.4        748.9
  594.68       215.14       205.36       298.72       614.8
  333.18        84.99       400.13       528.43       689.18
  297.43       762.63       582.48       507.58       215.03
  863.95      1610.9       1580.47       820.78       101.58
  431.45      1043.77       230.09       778.69              nan
  764.96       260.14       234.43       312.74      1234.29
  667.4        173.18       612.88       448.9        363.25
  151.96       206.35       400.91       464.72       521.94
  950.52      1203.4         99.59       525.33      1516.13
  826.51        44.71       829.51       733.58       407.54
  770.89       312.7350159  270.2999878  667.4000244]


Features Name : standard_cost
  388.920000    465
954.820000      396
53.620000       274
161.600000      235
260.140000      233
               ...
151.960000      124
206.350000      114
312.735016        1
270.299988        1
667.400024        1
Name: standard_cost, Length: 103, dtype: int64


Features Name : product_first_sold_date
 [41245. 41701. 36361. 36145. 42226. 39031. 34165. 39915. 33455. 38216.
 40784. 42172. 34527. 34586. 38193. 37873. 38206. 33888. 37337. 36334.
 42145. 42404. 34079. 41047. 42560. 42710. 41922. 37539. 42688. 38991.
 38647. 37874. 34996. 33549. 38693. 37668. 41533. 41009. 40553. 39427.
 38482. 35470. 41434. 36367. 38750. 41848. 34244. 42696. 38258. 41167.
 40672. 35707. 42295. 33552. 35667. 33879. 40670. 37626. 38339. 40303.
 34143. 35160. 36668. 36498. 34071. 40649. 37823. 36146. 42105. 34115.
 35052. 33364. 42218. 41345. 33429. 38859.    nan 36833. 37499. 41064.
 33259. 35560. 37838. 37698. 35378. 38573. 38002. 39526. 39880. 40487.
 40336. 40618. 34170. 40410. 42458. 39298. 35455. 37220. 37659. 40779.
 34556.]


Features Name : product_first_sold_date
  33879.0    234
41064.0      229
37823.0      227
39880.0      222
38216.0      220
            ...
41848.0      169
42404.0      168
41922.0      166
37659.0      163
34586.0      162
Name: product_first_sold_date, Length: 100, dtype: int64
```

In [12]:
```
# Separate lists of numeric and categorical columns
num_features=[cols for cols in df1.columns if df1[cols].dtype != 'O']
cat_features=[cols for cols in df1.columns if df1[cols].dtype == 'O']

print("Data has {} numerical columns: {}".format(len(num_features),num_features))
print("Data has {} Categorical columns : {}".format(len(cat_features),cat_features))
```

```
Data has 8 numerical columns: ['transaction_id', 'product_id', 'customer_id', 'transaction_date', 'online_order', 'list_price', 'standard_cost', 'product_first_sold_date']
Data has 5 Categorical columns : ['order_status', 'brand', 'product_line', 'product_class', 'product_size']
```

In [13]:
```
df1['online_order'].value_counts()
```

Out[13]:
```
1.0    9829
0.0    9811
Name: online_order, dtype: int64
```

In [14]:
```
#fill the missing data using the back fill method
df1['online_order']=df1['online_order'].fillna(method='bfill')
```

In [15]:
```
df1['online_order'].value_counts()
```

Out[15]:
```
1.0    10038
0.0     9962
Name: online_order, dtype: int64
```

In [16]:
```
#count the number of different brands
df1['brand'].value_counts()
```

Out[16]:
```
Solex             4253
Giant Bicycles    3312
WeareA2B          3295
OHM Cycles        3043
Trek Bicycles     2990
Norco Bicycles    2910
Name: brand, dtype: int64
```

In [17]:
```
# fill the missing data using the format fill method
df1['brand']=df1['brand'].fillna(method='ffill')
```

In [18]:
```
df1['product_line'].value_counts()
```

Out[18]:
```
Standard    14176
Road         3970
Touring      1234
Mountain      423
Name: product_line, dtype: int64
```

In [19]:
```
# fill the missing data using the back fill method
df1['product_line']=df1['product_line'].fillna(method='bfill')
```

In [20]:
```
# Count the number of differnt product classes
df1['product_class'].value_counts()
```

Out[20]:
```
medium    13826
high       3013
low        2964
Name: product_class, dtype: int64
```

In [21]:
```
#fill the missing data using the back-fill method
df1['product_class']=df1['product_class'].fillna(method='bfill')
```

In [22]:
```
df1['product_size'].value_counts()
```

Out[22]:
```
medium    12990
large      3976
small      2837
Name: product_size, dtype: int64
```

In [23]:
```
df1['product_size']=df1['product_size'].fillna(method='ffill')
```

In [24]:
```
df3 = df1.sort_values(['customer_id', 'transaction_date'])
```

In [25]:
```
# work out the average cost(to 2 decimical point) for each customer
pd.options.display.float_format='{:,.2f}'.format
customerStandardCost=df1.groupby(['customer_id']).mean().round(decimals=2)
customerStandardCost
```

Out[25]:

customer_id	transaction_id	product_id	online_order	list_price	standard_cost	product_first_sold_date
1	11,485.64	34.18	0.55	825.86	551.49	37,314.09
2	8,471.67	37.33	0.33	1,383.02	640.94	38,775.67
3	13,842.62	60.38	0.25	1,236.03	815.68	39,350.25
4	13,544.50	78.50	0.50	523.86	413.58	36,008.50
5	7,969.67	48.17	0.33	983.87	584.71	37,361.00
...	...	...	...	...	...	...
3497	8,565.00	32.33	0.67	1,248.02	698.58	38,680.33
3498	9,327.50	72.17	0.83	862.84	338.29	37,723.67
3499	6,871.29	42.71	0.57	1,096.21	388.32	38,163.86
3500	10,076.00	40.00	0.33	820.40	522.76	36,432.50
5034	14,292.67	0.00	0.00	506.64	416.81	37,254.67

3494 rows × 6 columns

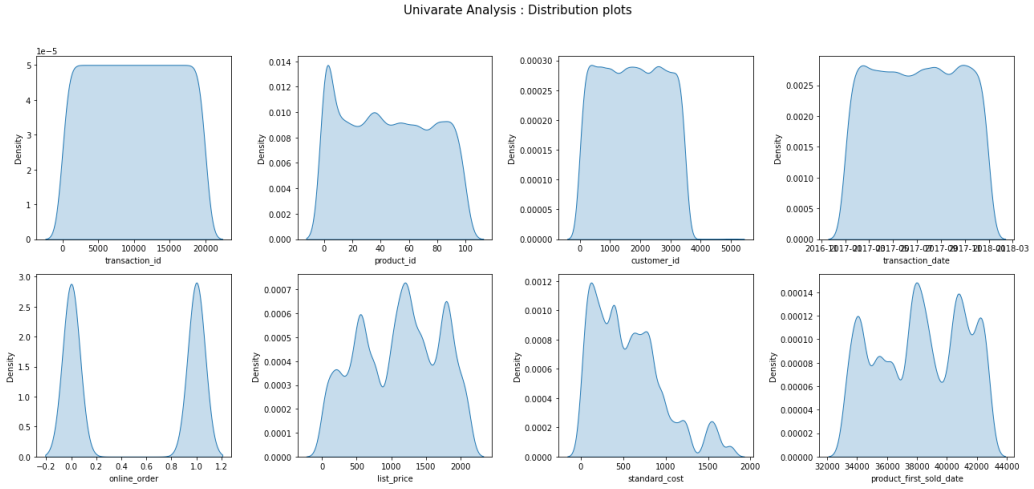
In [26]:
```
df3['standard_cost'] = df3['standard_cost'].fillna(customerStandardCost.standard_cost)
```

In [27]:
```
# fill the missing data with the average first sold date
df3['product_first_sold_date'] = df3['product_first_sold_date'].fillna(df3['product_first_sold_date'])
```

In [28]:
```
df1.info()
```

Out[28]:
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   transaction_id           20000 non-null  int64
 1   product_id               20000 non-null  int64
 2   customer_id              20000 non-null  int64
 3   transaction_date         20000 non-null  datetime64[ns]
 4   online_order             20000 non-null  float64
 5   order_status             20000 non-null  object
 6   brand                    20000 non-null  object
 7   product_line             20000 non-null  object
 8   product_class            20000 non-null  object
 9   product_size             20000 non-null  object
 10  list_price               20000 non-null  float64
 11  standard_cost            19803 non-null  float64
 12  product_first_sold_date  19803 non-null  float64
dtypes: datetime64[ns](1), float64(4), int64(3), object(5)
memory usage: 2.0+ MB
```

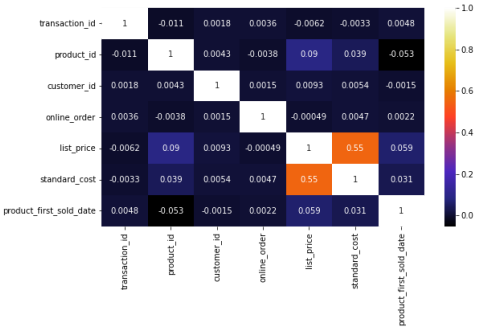
In [29]:
```
# Univariate Analysis:
# This Analysis is done for considering single variable and is the simplest form of analysis.

# Numerical Variable
plt.figure(figsize=(18,12))
plt.suptitle("Univarate Analysis : Distribution plots\n\n",fontsize=15)
for i in range(0,len(num_features)):
    plt.subplot(3,4,i+1)
    sns.kdeplot(x=df1[num_features[i]],fill=True)
    plt.xlabel(num_features[i])
    plt.tight_layout()
```



In [ ]:

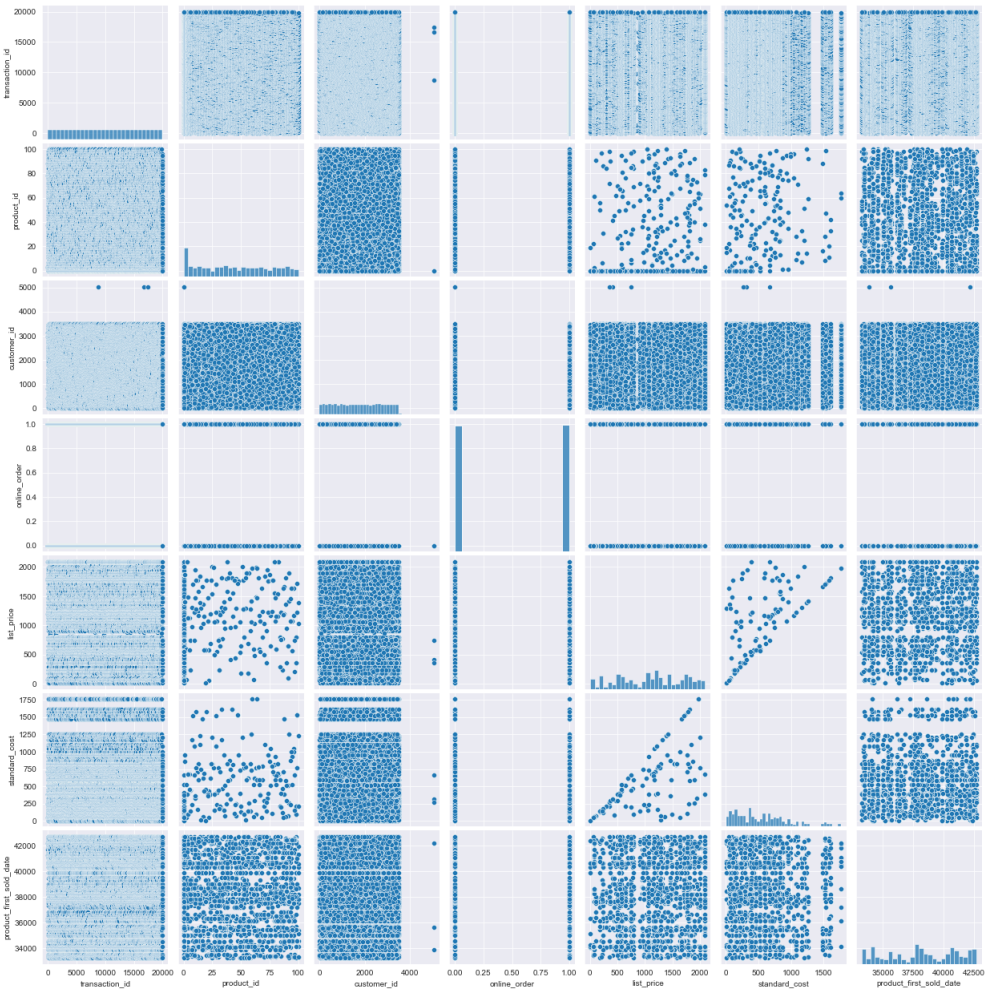
In [33]:
```
plt.figure(figsize=(9,5))
sns.heatmap(df1.corr(),cmap="CMRmap",annot=True)
```

Out[33]:
```
<AxesSubplot:>
```



In [34]:
```
sns.set_style(style='darkgrid')
sns.pairplot(df1)
```

Out[34]:
```
<seaborn.axisgrid.PairGrid at 0x2a14e216f40>
```



Customer Address

In [42]:
```
# read in the excel datasets and put into a panda dataframe
xls = pd.ExcelFile('KPMG_VI_New_raw_data_update_final.xlsx')
pd2 = pd.read_excel(xls, sheet_name=4, header=1)
```

In [43]:
```
df2 = pd.DataFrame(pd2)
```

In [48]:
```
# check out datatypes, columns name, counts
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3999 entries, 0 to 3998
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   customer_id          3999 non-null   int64 
 1   address              3999 non-null   object
 2   postcode             3999 non-null   int64 
 3   state                3999 non-null   object
 4   country              3999 non-null   object
 5   property_valuation   3999 non-null   int64 
dtypes: int64(3), object(3)
memory usage: 187.6+ KB
```

In [49]:
```
# check the first 5 records
df2.head()
```

Out[49]:

	customer_id	address	postcode	state	country	property_valuation
0	1	060 Morning Avenue	2016	New South Wales	Australia	10
1	2	6 Meadow Vale Court	2153	New South Wales	Australia	10
2	4	0 Holy Cross Court	4211	QLD	Australia	9
3	5	17979 Del Mar Point	2448	New South Wales	Australia	4
4	6	9 Oakridge Court	3216	VIC	Australia	9

In [50]:
```
# check the last 5 records
df2.tail()
```

Out[50]:

	customer_id	address	postcode	state	country	property_valuation
3994	3999	1482 Hauk Trail	3064	VIC	Australia	3
3995	4000	57042 Village Green Point	4511	QLD	Australia	6
3996	4001	87 Crescent Oaks Alley	2756	NSW	Australia	10
3997	4002	8194 Lien Street	4032	QLD	Australia	7
3998	4003	320 Acker Drive	2251	NSW	Australia	7

In [51]:
```
# check for null values
df2.isnull().sum()
```

Out[51]:
```
customer_id           0
address               0
postcode              0
state                 0
country               0
property_valuation    0
dtype: int64
```

In [52]:
```
# check that the customer id is unique
pd.Series(df2['customer_id']).is_unique
```

Out[52]:
```
True
```

In [53]:
```
# check there are no duplicate properties
df2.duplicated(subset=['address', 'postcode'], keep='first').value_counts()
```

Out[53]:
```
False    3999
dtype: int64
```

In [54]:
```
# show the frequency of different postcodes
df2['postcode'].value_counts()
```

Out[54]:
```
2170    31
2155    30
2145    30
2153    29
3977    26
        ..
3808     1
3114     1
4721     1
4799     1
3089     1
Name: postcode, Length: 873, dtype: int64
```

In [55]:
```
# show the frequency of different states
df2['state'].value_counts()
```

Out[55]:
```
NSW                2054
VIC                 939
QLD                 838
New South Wales      86
Victoria             82
Name: state, dtype: int64
```

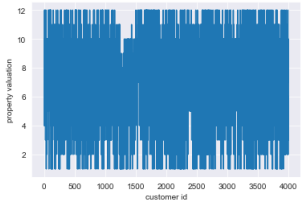
In [56]:
```
# replace all the misspelt or differently formatted data
df2['state'].replace({'New South Wales' : 'NSW', 'Victoria' : 'VIC'}, inplace=True)
```

In [57]:
```
# show the frequency of country to which the property belong to
df2['country'].value_counts()
```

Out[57]:
```
Australia    3999
Name: country, dtype: int64
```

In [58]:
```
# visualise the valuation prices of properties
x = df2['customer_id']
y = df2['property_valuation']

plt.plot(x, y)
plt.xlabel('customer id')
plt.ylabel('property valuation')
```

Out[58]:
```
Text(0, 0.5, 'property valuation')
```



In [59]:
```
# checking that the column names are in the correct format
df2.columns = map(str.lower, df2.columns)
df2.columns = map(str.strip, df2.columns)
```

In [60]:
```
df2
```

Out[60]:

	customer_id	address	postcode	state	country	property_valuation
0	1	060 Morning Avenue	2016	NSW	Australia	10
1	2	6 Meadow Vale Court	2153	NSW	Australia	10
2	4	0 Holy Cross Court	4211	QLD	Australia	9
3	5	17979 Del Mar Point	2448	NSW	Australia	4
4	6	9 Oakridge Court	3216	VIC	Australia	9
...	...	...	...	...	...	...
3994	3999	1482 Hauk Trail	3064	VIC	Australia	3
3995	4000	57042 Village Green Point	4511	QLD	Australia	6
3996	4001	87 Crescent Oaks Alley	2756	NSW	Australia	10
3997	4002	8194 Lien Street	4032	QLD	Australia	7
3998	4003	320 Acker Drive	2251	NSW	Australia	7

3999 rows × 6 columns

In [61]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3999 entries, 0 to 3998
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   customer_id         3999 non-null   int64
 1   address             3999 non-null   object
 2   postcode            3999 non-null   int64
 3   state               3999 non-null   object
 4   country             3999 non-null   object
 5   property_valuation  3999 non-null   int64
dtypes: int64(3), object(3)
memory usage: 187.6+ KB
```

Customer Demographic

In [68]:
```
# read in the excel datasets and put into a panda dataframe
xls = pd.ExcelFile('KPMG_VI_New_raw_data_update_final.xlsx')
pd1 = pd.read_excel(xls, sheet_name=3, header=1)
```

```
C:\Users\gmpl\AppData\Local\Temp\ipykernel_12788\1525478345.py:3: FutureWarning: Inferring datetime64[ns] from data containing strings is deprecated and will be removed in a future version. To retain the old behavior explicitly pass
Series(data, dtype=datetime64[ns])
  pd1 = pd.read_excel(xls, sheet_name=3, header=1)
```

In [69]:
```
df1 = pd.DataFrame(pd1)
```

In [70]:
```
# check out datatypes, columns name, counts
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 13 columns):
 #   Column                               Non-Null Count  Dtype
---  ------                               --------------  -----
 0   customer_id                          4000 non-null   int64
 1   first_name                           4000 non-null   object
 2   last_name                            3875 non-null   object
 3   gender                               4000 non-null   object
 4   past_3_years_bike_related_purchases  4000 non-null   int64
 5   DOB                                  3913 non-null   datetime64[ns]
 6   job_title                            3494 non-null   object
 7   job_industry_category                3344 non-null   object
 8   wealth_segment                       4000 non-null   object
 9   deceased_indicator                   4000 non-null   object
 10  default                              3698 non-null   object
 11  owns_car                             4000 non-null   object
 12  tenure                               3913 non-null   float64
dtypes: datetime64[ns](1), float64(1), int64(2), object(9)
memory usage: 406.4+ KB
```

In [71]:
```
# check the first 5 records
df1.head()
```

Out[71]:

	customer_id	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job_title	job_industry_category	wealth_segment	deceased_indicator	default	owns_car	tenure
0	1	Laraine	Medendorp	F	93	1953-10-12	Executive Secretary	Health	Mass Customer	N	"'	Yes	11.00
1	2	Eli	Bockman	Male	81	1980-12-16	Administrative Officer	Financial Services	Mass Customer	N	<script>alert('hi')</script>	Yes	16.00
2	3	Arlin	Dearle	Male	61	1954-01-20	Recruiting Manager	Property	Mass Customer	N	2018-02-01 00:00:00	Yes	15.00
3	4	Talbot	NaN	Male	33	1961-10-03	NaN	IT	Mass Customer	N	0 { _; } >_[(0)] { touch /tmp/blns.shellsh...	No	7.00
4	5	Sheila-kathryn	Calton	Female	56	1977-05-13	Senior Editor	NaN	Affluent Customer	N	NIL	Yes	8.00

In [72]:
```
# check for null values
df1.isnull().sum()
```

Out[72]:
```
customer_id                            0
first_name                             0
last_name                            125
gender                                 0
past_3_years_bike_related_purchases    0
DOB                                   87
job_title                            506
job_industry_category                656
wealth_segment                         0
deceased_indicator                     0
default                              302
owns_car                               0
tenure                                87
dtype: int64
```

In [73]:
```
# drop the column if too many datapoint is missing to be usable
drop_threshold = df1.shape[0]*0.5
df1 = df1.dropna(thresh=drop_threshold, how='all', axis='columns').copy()
```

In [74]:
```
# sanity check for duplicate data
column_names = df1.columns

for i in column_names:
  print((i, df1[i].is_unique))
```

```
('customer_id', True)
('first_name', False)
('last_name', False)
('gender', False)
('past_3_years_bike_related_purchases', False)
('DOB', False)
('job_title', False)
('job_industry_category', False)
('wealth_segment', False)
('deceased_indicator', False)
('default', False)
('owns_car', False)
('tenure', False)
```

In [75]:
```
# drop the column with corrupted data
del df1['default']
```

In [76]:
```
# make sure the ID only contain legitimate digits
df1['customer_id'] = df1['customer_id'].astype('int64')
```

In [77]:
```
# make sure that gender is a category
df1['gender'] = df1['gender'].astype('category')
```

In [78]:
```
# check the entered data
df1['gender'].value_counts()
```

Out[78]:
```
Female    2037
Male      1872
U           88
F            1
Femal        1
M            1
Name: gender, dtype: int64
```

In [80]:
```
# replace all the misspelt or differently formatted data
df1['gender'].replace({'M' : 'Male', 'F' : 'Female', 'Femal' : 'Female'}, inplace=True)
```

In [81]:
```
# check the frequency of different job titles
df1['job_title'].value_counts()
```

Out[81]:
```
Business Systems Development Analyst    45
Tax Accountant                          44
Social Worker                           44
Internal Auditor                        42
Recruiting Manager                      41
                                        ..
Database Administrator I                 4
Health Coach I                           3
Health Coach III                         3
Research Assistant III                   3
Developer I                              1
Name: job_title, Length: 195, dtype: int64
```

In [82]:
```
# fill the missing datapoint using the backward filling method
df1['job_title']= df1['job_title'].fillna(method='bfill')
```

In [83]:
```
# check the frequency of job industries
df1['job_industry_category'].value_counts()
```

Out[83]:
```
Manufacturing         799
Financial Services    774
Health                602
Retail                358
Property              267
IT                    223
Entertainment         136
Argiculture           113
Telecommunications     72
Name: job_industry_category, dtype: int64
```

In [84]:
```
# fill the missing datapoint using the forward filling method
df1['job_industry_category']= df1['job_industry_category'].fillna(method='ffill')
```

In [86]:
```
# make sure all dates are in the correct
df1['DOB'] = df1['DOB'].dt.date
```

In [87]:
```
# convert to boolean data type
f = {'N': False, 'Y': True}
df1['deceased_indicator'] = df1['deceased_indicator'].map(f).fillna(df1['deceased_indicator'])
```

In [88]:
```
# convert to boolean data type
g = {'No': False, 'Yes': True}
df1['owns_car'] = df1['owns_car'].map(g).fillna(df1['owns_car'])
```

In [89]:
```
# calculate the median value for tenure
np.nanmedian(df1['tenure'])
```

Out[89]:
```
11.0
```

In [90]:
```
# filling in the missing value for tenure
df1['tenure'].fillna(value = 11.0, inplace = True)
```

In [91]:
```
#check the dataset after cleaning
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 12 columns):
 #   Column                               Non-Null Count  Dtype
---  ------                               --------------  -----
 0   customer_id                          4000 non-null   int64
 1   first_name                           4000 non-null   object
 2   last_name                            3875 non-null   object
 3   gender                               4000 non-null   category
 4   past_3_years_bike_related_purchases  4000 non-null   int64
 5   DOB                                  3913 non-null   object
 6   job_title                            4000 non-null   object
 7   job_industry_category                4000 non-null   object
 8   wealth_segment                       4000 non-null   object
 9   deceased_indicator                   4000 non-null   bool
 10  owns_car                             4000 non-null   bool
 11  tenure                               4000 non-null   float64
dtypes: bool(2), category(1), float64(1), int64(2), object(6)
memory usage: 293.2+ KB
```

In [92]:
```
df1
```

Out[92]:

	customer_id	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job_title	job_industry_category	wealth_segment	deceased_indicator	owns_car	tenure
0	1	Laraine	Medendorp	Female	93	1953-10-12	Executive Secretary	Health	Mass Customer	False	True	11.00
1	2	Eli	Bockman	Male	81	1980-12-16	Administrative Officer	Financial Services	Mass Customer	False	True	16.00
2	3	Arlin	Dearle	Male	61	1954-01-20	Recruiting Manager	Property	Mass Customer	False	True	15.00
3	4	Talbot	NaN	Male	33	1961-10-03	Senior Editor	IT	Mass Customer	False	False	7.00
4	5	Sheila-kathryn	Calton	Female	56	1977-05-13	Senior Editor	IT	Affluent Customer	False	True	8.00
...	...	...	...	...	...	...	...	...	...	...	...	...
3995	3996	Rosalia	Halgarth	Female	8	1975-08-09	VP Product Management	Health	Mass Customer	False	False	19.00
3996	3997	Blanch	Nisuis	Female	87	2001-07-13	Statistician II	Manufacturing	High Net Worth	False	True	1.00
3997	3998	Sarene	Woolley	U	60	NaT	Assistant Manager	IT	High Net Worth	False	False	11.00
3998	3999	Patrizius	NaN	Male	11	1973-10-24	Software Engineer IV	Manufacturing	Affluent Customer	False	True	10.00
3999	4000	Kippy	Oldland	Male	76	1991-11-05	Software Engineer IV	Manufacturing	Affluent Customer	False	False	11.00

4000 rows × 12 columns

In [93]:
```
#print to csv file after the cleaning
df1.to_csv('./data1_cleaned.csv',index=False)
```

NewCustomerlist

In [103…
```
# read in the excel datasets and put into a panda dataframe
xls = pd.ExcelFile('KPMG_VI_New_raw_data_update_final.xlsx')
pd4 = pd.read_excel(xls, sheet_name=2, header=1)
```

```
C:\Users\gmpl\AppData\Local\Temp\ipykernel_12788\3341607728.py:3: FutureWarning: Inferring datetime64[ns] from data containing strings is deprecated and will be removed in a future version. To retain the old behavior explicitly pass Series(data, dtype=datetime64[ns])
  pd4 = pd.read_excel(xls, sheet_name=2, header=1)
```

In [104…
```
df4 = pd.DataFrame(pd4)
```

In [105…
```
# check out datatypes, columns name, counts
df4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 23 columns):
 #   Column                               Non-Null Count  Dtype
---  ------                               --------------  -----
 0   first_name                           1000 non-null   object
 1   last_name                            971 non-null    object
 2   gender                               1000 non-null   object
 3   past_3_years_bike_related_purchases  1000 non-null   int64
 4   DOB                                  983 non-null    datetime64[ns]
 5   job_title                            894 non-null    object
 6   job_industry_category                835 non-null    object
 7   wealth_segment                       1000 non-null   object
 8   deceased_indicator                   1000 non-null   object
 9   owns_car                             1000 non-null   object
 10  tenure                               1000 non-null   int64
 11  address                              1000 non-null   object
 12  postcode                             1000 non-null   int64
 13  state                                1000 non-null   object
 14  country                              1000 non-null   object
 15  property_valuation                   1000 non-null   int64
 16  Unnamed: 16                          1000 non-null   float64
 17  Unnamed: 17                          1000 non-null   float64
 18  Unnamed: 18                          1000 non-null   float64
 19  Unnamed: 19                          1000 non-null   float64
 20  Unnamed: 20                          1000 non-null   int64
 21  Rank                                 1000 non-null   int64
 22  Value                                1000 non-null   float64
dtypes: datetime64[ns](1), float64(5), int64(6), object(11)
memory usage: 179.8+ KB
```

In [106…
```
df4.head()
```

Out[106]:

	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job_title	job_industry_category	wealth_segment	deceased_indicator	owns_car	...	state	country	property_valuation	Unnamed: 16	Unnamed: 17	Unnamed: 18	Unnamed: 19	Unnamed: 20	Rank	Value
0	Chickie	Brister	Male	86	1957-07-12	General Manager	Manufacturing	Mass Customer	N	Yes	...	QLD	Australia	6	0.56	0.70	0.88	0.74	1	1	1.72
1	Morly	Genery	Male	69	1970-03-22	Structural Engineer	Property	Mass Customer	N	No	...	NSW	Australia	11	0.89	0.89	1.11	0.95	1	1	1.72
2	Ardelis	Forrester	Female	10	1974-08-28	Senior Cost Accountant	Financial Services	Affluent Customer	N	No	...	VIC	Australia	5	1.01	1.01	1.01	1.01	1	1	1.72
3	Lucine	Stutt	Female	64	1979-01-28	Account Representative III	Manufacturing	Affluent Customer	N	Yes	...	QLD	Australia	1	0.87	1.09	1.09	1.09	4	4	1.70
4	Melinda	Hadlee	Female	34	1965-09-21	Financial Analyst	Financial Services	Affluent Customer	N	No	...	NSW	Australia	9	0.52	0.52	0.65	0.65	4	4	1.70

5 rows × 23 columns

In [107…
```
df4.tail()
```

Out[107]:

	first_name	last_name	gender	past_3_years_bike_related_purchases	DOB	job_title	job_industry_category	wealth_segment	deceased_indicator	owns_car	...	state	country	property_valuation	Unnamed: 16	Unnamed: 17	Unnamed: 18	Unnamed: 19	Unnamed: 20	Rank	Val
995	Ferdinand	Romanetti	Male	60	1959-10-07	Paralegal	Financial Services	Affluent Customer	N	No	...	NSW	Australia	7	0.79	0.79	0.79	0.79	996	996	0.
996	Burk	Wortley	Male	22	2001-10-17	Senior Sales Associate	Health	Mass Customer	N	No	...	NSW	Australia	10	0.76	0.76	0.95	0.81	997	997	0.
997	Melloney	Temby	Female	17	1954-10-05	Budget/Accounting Analyst IV	Financial Services	Affluent Customer	N	Yes	...	QLD	Australia	2	0.85	1.06	1.06	1.06	997	997	0.
998	Dickie	Cubbini	Male	30	1952-12-17	Financial Advisor	Financial Services	Mass Customer	N	Yes	...	QLD	Australia	2	1.09	1.36	1.36	1.16	997	997	0.
999	Sylas	Duffill	Male	56	1955-10-02	Staff Accountant IV	Property	Mass Customer	N	Yes	...	NSW	Australia	9	0.47	0.59	0.73	0.62	1000	1000	0.

5 rows × 23 columns

In [108…
```
df4.isnull().sum()
```

Out[108]:
```
first_name                           0
last_name                           29
gender                               0
past_3_years_bike_related_purchases  0
DOB                                 17
job_title                          106
job_industry_category              165
wealth_segment                       0
deceased_indicator                   0
owns_car                             0
tenure                               0
address                              0
postcode                             0
state                                0
country                              0
property_valuation                   0
Unnamed: 16                          0
Unnamed: 17                          0
Unnamed: 18                          0
Unnamed: 19                          0
Unnamed: 20                          0
Rank                                 0
Value                                0
dtype: int64
```

In [109…
```
# drop the unnamed columns with unknown datapoints
del df4['Unnamed: 16']
del df4['Unnamed: 17']
del df4['Unnamed: 18']
del df4['Unnamed: 19']
del df4['Unnamed: 20']
```

In [110…
```
# sort by first name, then fill the missing last name
df4['last_name'] = df4.groupby('first_name').last_name.bfill().ffill()
```

In [111…
```
# check the entered data
df4['gender'].value_counts()
```

Out[111]:
```
Female    513
Male      470
U          17
Name: gender, dtype: int64
```

In [112…
```
# check the entered data
df4['state'].value_counts()
```

Out[112]:
```
NSW    506
VIC    266
QLD    228
Name: state, dtype: int64
```

In [113…
```
# check the entered data
df4['country'].value_counts()
```

Out[113]:
```
Australia    1000
Name: country, dtype: int64
```

In [114…
```
# check the entered data
df4['address'].nunique()
```

Out[114]:
```
1000
```

In [115…
```
# make sure all dates are in the correct format
df4['DOB'] = df4['DOB'].dt.date
df4['DOB']
```

Out[115]:
```
0      1957-07-12
1      1970-03-22
2      1974-08-28
3      1979-01-28
4      1965-09-21
          ...
995    1959-10-07
996    2001-10-17
997    1954-10-05
998    1952-12-17
999    1955-10-02
Name: DOB, Length: 1000, dtype: object
```

In [116…
```
# fill the missing date of birth with a random choice of date from the existing datapoints
df4['DOB'].fillna(lambda x: np.random.choice(df4['DOB']), inplace=True)
```

In [117…
```
# check the frequency of different job titles
df4['job_title'].value_counts()
```

Out[117]:
```
Associate Professor         15
Environmental Tech          14
Software Consultant         14
Chief Design Engineer       13
Assistant Manager           12
                            ..
Accountant II                1
Programmer IV                1
Administrative Officer       1
Accounting Assistant III     1
Web Developer I              1
Name: job_title, Length: 184, dtype: int64
```

In [118…
```
# fill the missing datapoint using the backward filling method
df4['job_title']= df4['job_title'].fillna(method='bfill')
```

In [119…
```
# check the frequency of job industries
df4['job_industry_category'].value_counts()
```

Out[119]:
```
Financial Services    203
Manufacturing         199
Health                152
Retail                 78
Property               64
IT                     51
Entertainment          37
Argiculture            26
Telecommunications     25
Name: job_industry_category, dtype: int64
```

In [120…
```
# fill the missing datapoint using the forward filling method
df4['job_industry_category']= df4['job_industry_category'].fillna(method='ffill')
```

In [121…
```
# make sure that the column names are in the correct format
df4.columns = map(str.lower, df4.columns)
df4.columns = map(str.strip, df4.columns)
```

In [122…
```
# round the values to the correct format
df4['value'] = df4['value'].round(3)
df4
```

Out[122]:

	first_name	last_name	gender	past_3_years_bike_related_purchases	dob	job_title	job_industry_category	wealth_segment	deceased_indicator	owns_car	tenure	address	postcode	state	country	property_valuation	rank	value
0	Chickie	Brister	Male	86	1957-07-12	General Manager	Manufacturing	Mass Customer	N	Yes	14	45 Shopko Center	4500	QLD	Australia	6	1	1.72
1	Morly	Genery	Male	69	1970-03-22	Structural Engineer	Property	Mass Customer	N	No	16	14 Mccormick Park	2113	NSW	Australia	11	1	1.72
2	Ardelis	Forrester	Female	10	1974-08-28	Senior Cost Accountant	Financial Services	Affluent Customer	N	No	10	5 Colorado Crossing	3505	VIC	Australia	5	1	1.72
3	Lucine	Stutt	Female	64	1979-01-28	Account Representative III	Manufacturing	Affluent Customer	N	Yes	5	207 Annamark Plaza	4814	QLD	Australia	1	4	1.70
4	Melinda	Hadlee	Female	34	1965-09-21	Financial Analyst	Financial Services	Affluent Customer	N	No	19	115 Montana Place	2093	NSW	Australia	9	4	1.70
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	Ferdinand	Romanetti	Male	60	1959-10-07	Paralegal	Financial Services	Affluent Customer	N	No	9	2 Sloan Way	2200	NSW	Australia	7	996	0.37
996	Burk	Wortley	Male	22	2001-10-17	Senior Sales Associate	Health	Mass Customer	N	No	6	04 Union Crossing	2196	NSW	Australia	10	997	0.36
997	Melloney	Temby	Female	17	1954-10-05	Budget/Accounting Analyst IV	Financial Services	Affluent Customer	N	Yes	15	33475 Fair Oaks Junction	4702	QLD	Australia	2	997	0.36
998	Dickie	Cubbini	Male	30	1952-12-17	Financial Advisor	Financial Services	Mass Customer	N	Yes	19	57666 Victoria Way	4215	QLD	Australia	2	997	0.36
999	Sylas	Duffill	Male	56	1955-10-02	Staff Accountant IV	Property	Mass Customer	N	Yes	14	21875 Grover Drive	2010	NSW	Australia	9	1000	0.34

1000 rows × 18 columns

In [123…
```
df4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 18 columns):
 #   Column                               Non-Null Count  Dtype  
---  ------                               --------------  -----  
 0   first_name                           1000 non-null   object 
 1   last_name                            1000 non-null   object 
 2   gender                               1000 non-null   object 
 3   past_3_years_bike_related_purchases  1000 non-null   int64  
 4   dob                                  1000 non-null   object 
 5   job_title                            1000 non-null   object 
 6   job_industry_category                1000 non-null   object 
 7   wealth_segment                       1000 non-null   object 
 8   deceased_indicator                   1000 non-null   object 
 9   owns_car                             1000 non-null   object 
 10  tenure                               1000 non-null   int64  
 11  address                              1000 non-null   object 
 12  postcode                             1000 non-null   int64  
 13  state                                1000 non-null   object 
 14  country                              1000 non-null   object 
 15  property_valuation                   1000 non-null   int64  
 16  rank                                 1000 non-null   int64  
 17  value                                1000 non-null   float64
dtypes: float64(1), int64(5), object(12)
memory usage: 140.8+ KB
```

In [124…
```
#print to csv file after the cleaning
df4.to_csv('./data4_cleaned.csv',index=False)
```

In [ ]: